UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/585,292	07/05/2006	Rongzhen Yang	P20651	2366

59796 7590 02/23/2011

INTEL CORPORATION
c/o CPA Global
P.O. BOX 52050
MINNEAPOLIS, MN 55402

EXAMINER	
MITCHELL, JASON D	

ART UNIT	PAPER NUMBER
2193	

NOTIFICATION DATE	DELIVERY MODE
02/23/2011	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

heather.l.adamson@intel.com

Office Action Summary	Application No.	Applicant(s)	
	10/585,292	YANG ET AL.	
	Examiner	Art Unit	
	JASON MITCHELL	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 December 2010*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *12/2/10*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This action is in response to an amendment filed on 12/13/10.

Claims 1-20 are pending in this application.

Response to Arguments

35 USC 101 rejection of claims 9-16

The applicants' amendments to claim 9 are sufficient to overcome the previous 35 USC 101 rejection which is consequently withdrawn.

35 USC 112 2nd rejection of claims 8 and 16

The applicants' amendments to claims 9 and 16 are sufficient to overcome the previous 35 USC 112 2nd rejections which are consequently withdrawn

35 USC 103(a) rejection of claims 1-20

In the par. bridging pp. 11 and 12 the applicants assert:

> Nothing in Pettis suggests that the statistical information is collected for *individual instruction addresses*; rather, Pettis is concerned with "the frequency with which each basic block of said first set of basic blocks transfers control to each other basic block of said first set of basic blocks." (Pettis, column 2, lines 65-68) (emphasis in original)

The claims do not explicitly require that statistical information be collect for "individual instruction addresses". Instead, the claims only require that information is collected which *corresponds to a plurality of* instruction addresses". Accordingly, the claims do not require a "one-to-one" correspondence between the addresses and the

information, and further do not require that this data be collected for each instruction of the program.

As noted by the applicants, Pettis discloses collecting statistical information for a plurality of “basic blocks”. Those of ordinary skill in the art would have understood that a “basic block” consists of a number of individual instructions. Accordingly, information corresponding to a “basic block” also corresponds to each of the instructions of that basic block.

Further, while Pettis does not explicitly disclose identifying the basic blocks or component instructions by their respective "addresses", it is believed that those of ordinary skill in the art would have recognized the reference as implicitly disclosing this functionality. Specifically, Pettis discloses identifying basic blocks in the context of JUMP instructions (see e.g. col. 4, lines 43-47 “data specifying the source of the JUMP instruction, and a JUMP instruction to the beginning of the basic block"). Those of ordinary skill in the art would have understood that Pettis’ “JUMP instruction” describes an instruction indicating that control should be passed to the instruction or basic block at the address specified by the JUMP instruction. Accordingly, it should be seen that the "dump” information collected by Pettis consists of a list of JUMP instructions linked by there target addresses (see e.g. col. 4, lines 53-57 “the various jumps and the number of times each JUMP instruction was executed").

Still further, it is noted that many other references are available which describe generating similar "dump" information which explicitly disclose use of an address (see e.g. US 5,313,616 to Cline et al. col. 12, lines 33-43 “candidate block’s starting address

… destination address of the call or branch instruction"; WO 2006/092079 to Yang et al. Abstract "A dump that comprises the instructions and corresponding instruction addresses may also be obtained"). While the examiner believes the current rejection meets the broad scope of the claims (as discussed above), it is further noted that references such as Yang seem to address the spirit of the arguments presented with this amendment. Accordingly, it is believed that amendments to more narrowly claim the dump information would be addressed with such references.

In the 1st full par. on pg. 12, the applicants assert:

> Applicant believes this portion of Pettis to be describing source code and not, as now claimed, "dump information [that] was produced during execution of the software."

The examiner respectfully disagrees. Specifically, Pettis explicitly disclosed collecting the cited dump data at run time (see e.g. col. 4, lines 53-57 "After the code has been execute [sic] with the test data").

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1, 3-4, 7-9, 11-12, 15-17 and 19-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 5,212,794 to Pettis et al. (Pettis) in view of "A

Language Independent Approach for Detecting Duplicated Code" by Ducasse et al. (Ducasse).

Claims 1, 9 and 17: Pettis discloses a method comprising:

obtaining performance data for software that has executed in a data processing system, wherein the performance data comprises a plurality of instruction addresses and corresponding performance information for each instruction address of the plurality of instruction addresses (col. 2, lines 51-54 "statistical information obtained by running the computer code"; col. 4, lines 43-47 "storage space for a counter used to recorded the number of times the jump is executed, data specifying the source of the JUMP instruction, and a JUMP instruction to the beginning of the basic block"; here the disclosed "data specifying the source of the JUMP" at least broadly constitutes an address, further note that this data "corresponds" to each of the instructions in the basic block);

obtaining dump information from the data processing system, wherein the dump information was produced during execution of the software (col. 2, lines 51-54 "statistical information obtained by running the computer code"), and wherein the dump information comprises a plurality of instructions, with each instruction of the plurality of instructions having a corresponding instruction address of the plurality of instruction addresses (col. 4, lines 43-47 "record the number of times the jump is executed, data specifying the source of the JUMPO instruction, and a JUMP instruction to the beginning of the basic block");

automatically identifying common code segments in the performance data, wherein a common code segment comprises an ordered set of multiple instructions that appears multiple times in the performance data (col. 9, lines 38-47 “The linkage placement of the procedures is determined by constructing groups similar to the chains described above with reference to basic blocks … by first choosing the edge with the heaviest weight”; note that a procedure constitutes a set of ordered instructions and the disclosed grouping of procedures constitutes an identification of a superset of ordered instructions which appear frequently in the trace); and

generating aggregate performance data for the common code segments, based at least in part on the instruction addresses associated with the common code segments from the dump information, the instruction addresses from the performance data, and the corresponding performance information from the performance data (col. 9, lines 50-53 “The two nodes are merged into a single node or group having two procedures therein”; col. 9, lines 30-34 “If a procedure calls another from several different places within itself, ... those weights are first summed together to form the edge weight shown in the graph").

Pettis does not disclose automatically identifying common code segments in the dump information … and generating aggregate performance data for the common code segments in the dump information.

Ducasse teaches identifying common code segments in dump information (Abstract "detect a significant amount of code duplication").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Ducasse's techniques (Abstract "detect a significant amount of code duplication") to identify duplicate code segments in Pettis' dump information (col. 2, lines 59-61 "computer program comprising a set of basic blocks") and to aggregate data for these duplicated code sections (see e.g. Pettis "weights are first summed together to form the [aggregated] edge weight"). Those of ordinary skill in the art would have been motivated to do so because "[c]ode duplication increases the size of the code ... expanding the size of the executable" (see Ducasse pg. 1, col. 2, 1st partial par.) and Pettis attempts to reduce the size of 'frequently executed' code loaded in memory (see e.g. Pettis col. 7, lines 44-50 "minimize the size of the primary part of the procedures"). In other words, unrecognized code duplication could result in Pettis' system two or more separate chains (e.g. Clone1—P and Clone2—P) which would more optimally be represented as a single chain (e.g. Clone1/Clone2—P). This single chain representation would result in a smaller "primary part" of the program being loaded into memory and a more accurate weight being applied to the behavior so that the importance of this chain can more accurately be recognized. Further note that Ducasse explicitly suggest such a combination (pg. 9, col. 2, 1st partial par. "Investigate if textual comparison is a useful approach for exploring the structure of ... program execution traces, wherein it could be interesting to find recurring patterns of execution sequences")

Claims 3, 11 and 19: The rejections of claims 1, 9 and 17 are incorporated; further Ducasse teaches the operation of identifying common code segments in the dump information comprises:

selecting a candidate code segment from the dump information (pg. 3, col. 1, 1st par. "compare every entity … with every other entity");

determining whether the candidate code segment occurs multiple times in the dump information (pg. 3, col. 1, 1st par. "compare every entity … with every other entity"); and

identifying the candidate code segment as a common code segment in response to determining that the candidate code segment occurs multiple times in the dump information (pg. 3, col. 1, 1st par. "The result is a Boolean true for an exact match").

Claims 4 and 12: The rejections of claims 1, 9 are incorporated; further Ducasse teaches the operation of identifying common code segments in the dump information comprises:

selecting a candidate code segment from the dump information (pg. 3, col. 1, 1st par. "compare every entity … with every other entity");

determining whether the dump information includes at least one additional absolute match for the candidate code segment (pg. 3, col. 1, 1st par. "compare every entity … with every other entity"); and

identifying the candidate code segment as a common code segment in response to determining that the dump information includes at least one additional absolute match for the candidate code segment (pg. 3, col. 1, 1st par. “The result is a Boolean true for an exact match”).

Claims 7, 15 and 20: The rejections of claims 1, 9 and 17 are incorporated; further Pettis and Ducasse teach:

the operation of identifying common code segments in the dump information comprises identifying at least first and second common code segments (Ducasse pg. 3, col. 1, 1st par. “compare every entity … with every other entity”); and

the operation of generating aggregate performance data for the common code segments comprises:

collecting performance data for multiple instances of the first common code segment (Pettis col. 2, lines 51-54 “statistical information obtained by running the computer code");

generating aggregate performance data for the first common code segment, based at least in part on the performance data for the multiple instances of the first common code segment (Pettis col. 9, lines 50-53 “The two nodes are merged into a single node”);

collecting performance data for multiple instances of the second common code segment (Pettis col. 2, lines 51-54 “statistical information obtained by running the computer code"); and

generating aggregate performance data for the second common code segment, based at least in part on the performance data for the multiple instances of the second common code segment (Pettis col. 9, lines 50-53 “The two nodes are merged into a single node”).

Claims 8 and 16: The rejections of claims 7, 15 are incorporated; further Pettis discloses the operation of generating aggregate performance data for the common code segments comprises:

collecting performance information corresponding to instruction addresses for a plurality of instances of the common code segment in the dump information (col. 2, lines 51-54 “statistical information obtained by running the computer code").

Claims 2, 6, 10, 14 and 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 5,212,794 to Pettis et al. (Pettis) in view of “A Language Independent Approach for Detecting Duplicated Code” by Ducasse et al. (Ducasse) in view of .

Claims 2, 10 and 18: The rejections of claims 1, 9 and 17 are incorporated; further Pettis and Ducasse do not teach: obtaining performance data for instructions generated by a dynamic compiler; and generating aggregate performance data for common code segment generated by the dynamic compiler.

Chauvel teaches obtaining performance data for instructions generated by a dynamic compiler; and generating aggregate performance data for common code segment generated by the dynamic compiler (par. [0012] “acquiring a virtual machine profile that relates a performance characteristic to individual operations and generating an aggregate value for the performance characteristic based on the application profile and the virtual machine profile”).

It would have been obvious to one of ordinary skill in the art at the time the invention was made obtain and aggregate performance data as taught by Pettis and Ducasse (Pettis col. 2, lines 51-54 “statistical information obtained by running the computer code"; col. 9, lines 50-53 “The two nodes are merged into a single node or group having two procedures therein”) for instructions generated by a dynamic compiler as taught by Chauvel (par. [0012] “acquiring a virtual machine profile”). Those of ordinary skill in the art would have been motivated to do so in order to optimize an application compiled by a dynamic compiler (par. [0011] “a need has arisen for a method and apparatus for reliably estimating performance characteristics, ... in a device using a virtual machine interface").

Claims 6 and 14: The rejections of claims 1, 9 and are incorporated; further Pettis and Ducasse do not explicitly teach the performance information comprises one or more measurements selected from the group consisting of: execution time data for individual instructions; and cache miss data for individual instructions.

Chauvel teaches performance information comprises one or more measurements selected from the group consisting of: execution time data and cache miss data (par. [0025] “performance characteristics such as … cache misses … operation execution time”).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to gather performance information comprising execution time data or cache miss data as taught by Chauvel (par. [0025] “performance characteristics such as … cache misses … operation execution time”) in the performance information gathered Pettis (Pettis col. 2, lines 51-54 “statistical information obtained by running the computer code"). Those of ordinary skill in the art would have been motivated to do so would allow the tool to focus on the most problematic areas of the program (see e.g. col. 4, lines 30-32 "there are fewer cache misses"; Chauvel par. [0025] “cache misses”).

Claims 5 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 5,212,794 to Pettis et al. (Pettis) in view of “A Language Independent Approach for Detecting Duplicated Code” by Ducasse et al. (Ducasse) in view of “A Program for Identifying Duplicated Code” by Baker (Baker).

Claims 5 and 13: The rejections of claims 1, 9 and are incorporated; further Ducasse teaches the operation of identifying common code segments in the dump information comprises:

selecting a candidate code segment from the dump information (pg. 3, col. 1, 1st par. "compare every entity … with every other entity");

determining whether the dump information includes at least one additional match for the candidate code segment (pg. 3, col. 1, 1st par. "compare every entity … with every other entity"); and

identifying the candidate code segment as a common code segment in response to determining that the dump information includes at least one additional match for the candidate code segment (pg. 3, col. 1, 1st par. "The result is a Boolean true for an exact match").

Pettis and Ducasse do not teach identifying elements in the candidate code segment as significant; and wherein the additional match comprises instructions with elements matching the significant elements in the candidate code segment.

Baker teaches identifying elements in the candidate code segment as significant; and determining whether the dump information includes at least one additional match for the candidate code segment, wherein the additional match comprises instructions with elements matching the significant elements in the candidate code segment (pg. 2, par. bridging cols. 1 & 2 "the program can look for parameterized matches, where the code

sections match except for a one-to-one correspondence between candidates for parameters such as variables").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to substitute Baker's "parameterized" matching algorithm (pg. 2, par. bridging cols. 1 & 2 "the program can look for parameterized matches") for Decease's "absolute" matching algorithm (pg. 3, col. 1, 1st par. "The result is a Boolean true for an exact match"). Those of ordinary skill in the art would have been motivated to do so because parameterized matching will result in the identification of more duplication (pg. 3, col. 1, 1st full par. "code will have more parameterized matches than exact matches") and thus greater optimization (as discussed in the rejection of the parent claim(s)).

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to JASON MITCHELL whose telephone number is (571)272-3728. The examiner can normally be reached on Monday-Thursday and alternate Fridays 7:30-5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Bullock Lewis can be reached on (571) 272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Jason Mitchell/
Primary Examiner, Art Unit 2193